02

Estructura de excepciones

Transcripción

[00:00] Sean bienvenidos a una nueva clase de su curso de Java, entendiendo excepciones. Recapitulando un poco lo que hemos visto en la clase anterior, ya sabemos que crear nuestras propias bombas, mejor dicho, detonar nuestras propias bombas con throw. Yo sé que hasta ahora hemos ido tocando muchos tópicos.

[00:21] Por ejemplo ya hemos tocado lo que es la referencia, la pila de ejecución, hasta hemos explicado un poco de lo que es la memoria HEAP. Y bueno, si bien el tema de excepciones es un tema pues digamos un poco extenso, quizás hasta un poco complicado, no se preocupen porque lo vamos a aplicar de tal forma que va a ser muy sencillo de entender para ustedes.

[00:47] En esta parte ya vamos a tratar de ver cómo crear nuestras propias bombas ya personalizadas con nuestros errores que nosotros queremos lanzar. Y para esto vamos a hacer un pequeño review de lo que es el objeto excepción. Yo les dije que toda excepción es un objeto, entonces como objeto, yo debería ser capaz de entrar y ver sus propiedades. ¿Qué voy a hacer?

[01:13] Voy a apretar control y voy a irme a la clase ArithmeticException para explorar qué hay adentro de este objeto. Y vemos que llegué aquí a ArithmeticException.class, llegué aquí, tiene un constructor super. Él llega aquí, vamos desde el inicio.

[01:30] Tenemos la documentación de Java, el Javadoc como es llamado. Él aquí da una pequeña explicación de qué es lo que sucede con esta excepción, aqu

lo que él dice, por ejemplo es: "lanzada cuando una condición aritmética excepcional ha ocurrido", por ejemplo, un integer dividido entre 0, retorna una instancia de esta clase.

[01:52] Justamente fue lo que nosotros hemos provocado. Y aquí también da una explicación de cómo debería ser construido este objeto y sobre qué cuáles son las clases padre, todo, pero vamos con calma, vamos bajando por ahora. Vemos que ArithmeticException extiende the RuntimeException. Aquí ya vemos un uso claro de la herencia.

[02:17] ¿Cuáles eran los objetivos de la herencia? Establecer relación entre objetos y reutilización de código. Aquí ya vemos claramente que él extiende RuntimeException. Este private static final long serialVersion es un número largo que sirve pues digamos para serializar los objetos.

[02:38] Es un término que no viene al caso ahorita, serializar, pero solamente para quien tenga curiosidad es cuando yo voy a enviar objetos por internet, por la red, ellos tienen un número de serie único, para ser identificados por el receptor, por ahora no viene al caso. Si quieren, olviden lo que dije.

[02:55] Y aquí tenemos el primer constructor, un constructor sin parámetros por defecto, que llama un súper, ese súper lo que hace referencia a constructor de la clase padre, clase padre, ArithmeticaException. Y vemos que aquí también él tiene otro constructor, un segundo constructor con un String s que no sé qué es. Es un mensaje, acá la documentación dice que es un mensaje.

[03:22] Entonces yo puedo mandarle un mensaje al ArithmeticException aquí, y vimos que sí se puede, yo aquí puedo mandarle un mensaje, lo vimos en la clase pasada. Y él igual llama al constructor de la clase padre y le envía como parámetro ese string, esa clase padre ya tiene dos constructores entonces, y aquí acaba la clase.

[03:44] Bueno, ya exploramos ArithmeticException, vamos a explorar la clase padre. Entonces nuevamente control RuntimeException y vemos nuevamente,

bueno, acá el Java doc correspondiente, esta es la superclase de las excepciones que pueden ser lanzadas durante alguna operación normal, etcétera. Es lo que dice el Java doc.

[04:07] Y aquí tenemos que él extiende de exception, o sea, hasta ahora ya tenemos tres niveles: ArithmeticException, RuntimeException y Exception. Tres niveles, por el momento y nuevamente RuntimeException lanza un super aquí, o sea, él llama al constructor de su clase padre, que es Exception. Y aquí, de la misma forma, él recibe un mensaje y lanza ese mensaje para su clase padre.

[04:33] Y aquí tiene otros constructores también. Él tiene para lanzar un mensaje y tiene una clase Throwable cause. Esto es nuevo para nosotros. La verdad es que él tiene aún más métodos aquí, solamente para lanzar un Throwable, lo vamos a ver de aquí a poco. Y aquí tiene incluso para lanzar muchos más parámetros: enableSuppression, writableStackTrace.

[05:01] O sea, vemos que en realidad es una clase con un poco más de abstracción que las clases hijas. ¿Y en todos los casos, el común denominador cuál es? Super. Él siempre llama al método, a los constructores de la clase padre, que en este caso es Exception. Eso nos dice que Exception es una clase quizás hasta un poco más grande.

[05:25] Y, como no podía ser de otra forma, vamos a entrar a Exception. Y aquí finalmente, ya que llegamos a Exception, nuevamente tiene su su serial, y tenemos el consultor Exception. Vemos algo curioso: Exception extiende de Throwable, y esta es la razón por la cual vamos ahora a hacer el nexo entre esta palabra reservada de aquí y throwable.

[05:55] ¿Por qué nosotros no podíamos lanzar un objeto de clase cuenta? Porque solamente podemos lanzar objetos que extiendan de throwable, que throwable es la clase padre de las excepciones. Si entramos aquí, vamos a ver que él implementa, ojo, es diferente aquí, haciendo un repaso al curso

anterior, en la que hace un extend, él es hijo de throwable y él aquí implementa Serializable, la interfaz Serializable.

[06:28] Que como les dije anteriormente, sirve para lanzar objetos en la red en internet, para enviar objetos en internet. No viene al caso ahorita. Y aquí él tiene ya un montón de lógica, digamos, de campos. Tiene un una clase objeto backtrace, tiene mensaje detallado, un string. En fin, es una clase, digamos un poco extensa. Yo creo que es muy extensa.

[06:55] Y si le damos "Ctrl + O" vamos a ver que él aquí, vamos a darle más arriba, con "Ctrl + O" podemos ver una hacer un overview de la clase completa, de qué métodos tiene. Entonces vamos aquí, "Ctrl + O" y aquí vemos todo el detalle de métodos que él tiene y aquí tiene causa, stackTrace, depth, profundidad, throwable, ahí están todos los constructores que hemos ido viendo en las clases hijas.

[07:22] PrintStackTrace, en fin, una infinidad de métodos. Entonces, ahora, nuevamente volviendo a bajar, ya hemos subido hasta throwable. Hasta ahora tenemos cinco niveles de abstracción. O claro, cinco serían. Serían ArithmeticException, que extiende a RuntimeException, que a la vez extiende de Exception y de throwable. No, cuatro niveles, disculpa, hay cuatro niveles de abstracción.

[07:50] Entonces, volviendo a Exception, vemos ahí, pues que de igual forma, él tiene los constructores en todos casos llama a super, que es el constructor de la clase padre y listo. Entonces, ahora que ya conozco throwable, al mismo tiempo se los diversos constructores. ¿Por qué ArithmeticException tiene que extender de Runtime y Runtime de Exception, y Exception de throwable?

[08:18] No estoy consiguiendo entender eso. Vamos a explicarlo mejor en el siguiente video.